Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Complete The Look Recommendation with Street Fashion Images

Bhaskar Bagchi
bagchi.bhaskar@cse.iitkgp.ernet.in

October 21, 2018

Content

- Given an item(clothing) in the shopping cart the problem statement is to suggest items complementary to it which may contain garments or accessories which makes a complete set as per current fashion.

Introduction
Problem Definition

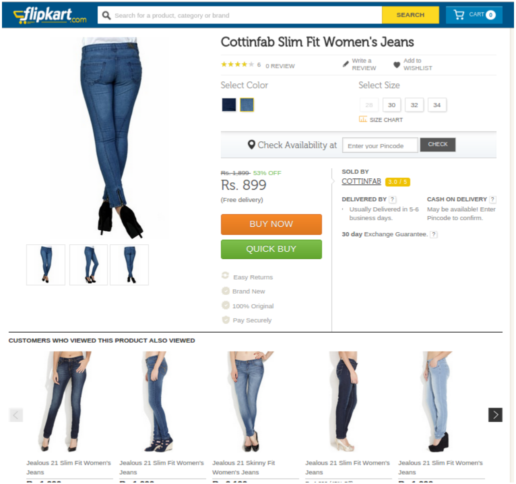


Figure: Existing Recommendation Systems

Introduction
Problem Definition

Figure: Visualization of the problem statement

Mathematical Formulation

Given an image i containing 'k' part–features, we describe the image P_i as $P_i^T := [p_{i1}, p_{i2}, ..., p_{ik}]$ where each p_{ij} are textual part–features, which are 2–tuples.

Mathematical Formulation

We learn a model from our dataset of fashion images, say **P**, where **P** := $[P_1, P_2, ...P_n]^T$.

Mathematical Formulation

The task of our recommendation system is, given one or more apparel, and corresponding part features *p*'s as input query, recommend garments which can be worn with it/them as a set.

Approach
Flow Diagram

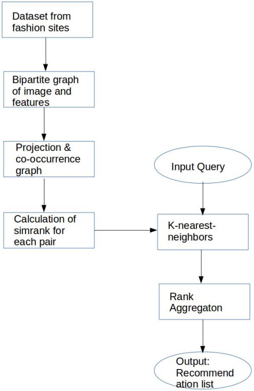


Figure: Flow Diagram of Proposed Approach

- Scraped more than 500 images of female fashionstas from `www.chictopia.com`. These images covered an appreciable range of street fashion from corporate dressing sense to the most casual of the dresses.

- Scraped more than 500 images of female fashionstas from `www.chictopia.com`. These images covered an appreciable range of street fashion from corporate dressing sense to the most casual of the dresses.
- Created a vocabulary of part features. Manually normalize the tags associated with each image.

- Scraped more than 500 images of female fashionstas from `www.chictopia.com`. These images covered an appreciable range of street fashion from corporate dressing sense to the most casual of the dresses.
- Created a vocabulary of part features. Manually normalize the tags associated with each image.
- Ended up with a codebook of total of *48* unique categories including garments like tops, jeans, etc. and accessories like watches, bracelets, etc. and *632* unique items i.e. category-description pair.

Bipartite Network and Co–occurrence Graph

- A bipartite graph is formed with dataset images P's as the first partite sets and part–features p_i's as the second partite set. There exists an edge between ever part feature and the image in which it occurred.

Bipartite Network and Co–occurrence Graph

- A bipartite graph is formed with dataset images P's as the first partite sets and part–features p_i's as the second partite set. There exists an edge between ever part feature and the image in which it occurred.
- The bipartite graph is then projected to the set of part features.

Bipartite Network and Co–occurrence Graph

- A bipartite graph is formed with dataset images *P*'s as the first partite sets and part–features p_i's as the second partite set. There exists an edge between ever part feature and the image in which it occurred.
- The bipartite graph is then projected to the set of part features.
- The projected graph so obtained is a weighted co–occurrence graph of the part features. Construction of this graph gives us the relation between different garments and accessories which can be used together and are complementary to each other.

Bipartite Network and Co–occurrence Graph

- A bipartite graph is formed with dataset images P's as the first partite sets and part–features p_i's as the second partite set. There exists an edge between ever part feature and the image in which it occurred.
- The bipartite graph is then projected to the set of part features.
- The projected graph so obtained is a weighted co–occurrence graph of the part features. Construction of this graph gives us the relation between different garments and accessories which can be used together and are complementary to each other.
- This step helps us learn a correlation and inter-dependence between various part features from the dataset.

- The co-occurrence graph falls in a domain where nodes represents the objects and edges represents the relations between them. We use *Simrank* to measure the similarity based on *structural context* of the graph.

- The co-occurrence graph falls in a domain where nodes represents the objects and edges represents the relations between them. We use *Simrank* to measure the similarity based on *structural context* of the graph.
- Convert the co–occurrence graph into a directed graph where each edge between part features p_a and p_b in the original graph is replaced by two directed edges $p_a \rightarrow p_b$ and $p_b \rightarrow p_a$ both with weights equal to the weight of original edge.

- The co-occurrence graph falls in a domain where nodes represents the objects and edges represents the relations between them. We use *Simrank* to measure the similarity based on *structural context* of the graph.
- Convert the co–occurrence graph into a directed graph where each edge between part features p_a and p_b in the original graph is replaced by two directed edges $p_a \rightarrow p_b$ and $p_b \rightarrow p_a$ both with weights equal to the weight of original edge.
- Compute *Simrank* between each pair of nodes.

- Given a part–feature *p* as query we locate the node corresponding to that part feature in the co–occurrence graph.

- Given a part–feature *p* as query we locate the node corresponding to that part feature in the co–occurrence graph.
- We find out other nodes which are close to it, i.e. nodes which have highest simrank value with this node.

Similarity Measure & Nearest Neighbor

Nearest Neighbor Consensus

- Given a part–feature *p* as query we locate the node corresponding to that part feature in the co–occurrence graph.
- We find out other nodes which are close to it, i.e. nodes which have highest simrank value with this node.
- The rationale behind this step is that since the graph had edges between part features that were used together by fashionistas and as the simrank values decrease with increase in node distances, the *k*–nearest–neighbors will be those part features which were frequently used with the selected item and are contemporary to it.

- Given a part–feature *p* as query we locate the node corresponding to that part feature in the co–occurrence graph.
- We find out other nodes which are close to it, i.e. nodes which have highest simrank value with this node.
- The rationale behind this step is that since the graph had edges between part features that were used together by fashionistas and as the simrank values decrease with increase in node distances, the *k*–nearest–neighbors will be those part features which were frequently used with the selected item and are contemporary to it.
- We get a list of k part features $p_1, p_2, ...p_k$ which are structurally close to the input feature and thus they can be recommended for the given query part feature.

- Say we have j part features $p_1, p_2, ...p_j$ as input query, we find out individual k–nearest–neighbors for each part feature.

- Say we have j part features $p_1, p_2, ...p_j$ as input query, we find out individual k–nearest–neighbors for each part feature.
- we have j ranked lists, each with k members, which are recommendation related to each input feature.

- Say we have j part features $p_1, p_2, ...p_j$ as input query, we find out individual k–nearest–neighbors for each part feature.
- we have j ranked lists, each with k members, which are recommendation related to each input feature.
- Assigns a score corresponding to position in which a part feature appears within each ranked list. In our case, for each list i, ${p_a}^i$ is assigned a weight ${B_{p_a}}^i = k$ * fraction of part features in the list appearing below p_a.

- Say we have j part features $p_1, p_2, ...p_j$ as input query, we find out individual k–nearest–neighbors for each part feature.
- we have j ranked lists, each with k members, which are recommendation related to each input feature.
- Assigns a score corresponding to position in which a part feature appears within each ranked list. In our case, for each list i, ${p_a}^i$ is assigned a weight ${B_{p_a}}^i = k$ * fraction of part features in the list appearing below p_a.
- The *Broda* score of each element B_{p_a} is the the sum of *Broda* scores for that part feature in all the lists.

Aggregating Ranked Item Recommendations
Rank Aggregation

- Say we have j part features $p_1, p_2, ...p_j$ as input query, we find out individual k–nearest–neighbors for each part feature.
- we have j ranked lists, each with k members, which are recommendation related to each input feature.
- Assigns a score corresponding to position in which a part feature appears within each ranked list. In our case, for each list i, ${p_a}^i$ is assigned a weight ${B_{p_a}}^i = k$ * fraction of part features in the list appearing below p_a.
- The *Broda* score of each element B_{p_a} is the the sum of *Broda* scores for that part feature in all the lists.
- We can recomment the top k elements from this ranked list to the user.

- We took 20 images as test set from our dataset. Since each image is user tagged, we have labelled ground truth for computing the required metrics.

- We took 20 images as test set from our dataset. Since each image is user tagged, we have labelled ground truth for computing the required metrics.
- For each image we took, we used all its part features individually as one feature input. We also used various permutations of 2 part features and 3 part features as input to the recommender and compared the recommended part features with the ground truth.

Experimental Results
Evaluation Methodology

- We took 20 images as test set from our dataset. Since each image is user tagged, we have labelled ground truth for computing the required metrics.
- For each image we took, we used all its part features individually as one feature input. We also used various permutations of 2 part features and 3 part features as input to the recommender and compared the recommended part features with the ground truth.
- Then we calculated *precision, recall* and *f1* values for 158 sets of recommendations.

- We took 20 images as test set from our dataset. Since each image is user tagged, we have labelled ground truth for computing the required metrics.
- For each image we took, we used all its part features individually as one feature input. We also used various permutations of 2 part features and 3 part features as input to the recommender and compared the recommended part features with the ground truth.
- Then we calculated *precision, recall* and *f1* values for 158 sets of recommendations.

Formula

$$precision = \frac{no\ of\ matched\ recommendations}{no\ of\ recommendations}$$

$$recall = \frac{no\ of\ matched\ recommendation}{no\ of\ items\ in\ actual\ image}$$

Experimental Results
Results

Out of the 158 recommendation sets that we tested, 53 were 1 part feature input, 54 were 2 part feature input and 51 as 3 part feature input. For each generated recommendations we calculated the precision and recall.

Table: Precision

No. of inputs	Max Precision	Avg Precision
1	1	0.31
2	0.75	0.31
3	0.6	0.28

Table: Recall

No. of inputs	Max Recall	Avg Recall
1	0.8	0.23
2	1	0.44
3	1	0.48

Table: f1 score

No. of inputs	Max f1	Min f1
1	0.89	0.13
2	0.71	0.1
3	0.67	0.1

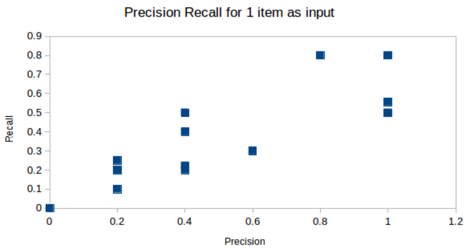
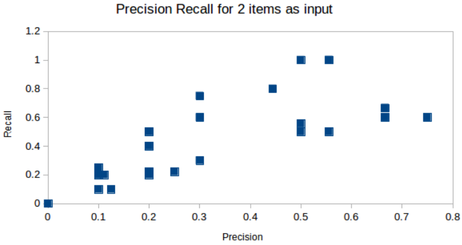


Figure: Precision-Recall for 1 item input

Experimental Results
Precision Recall Graphs



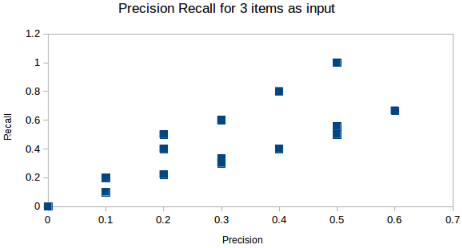


Table: User rating for recommendation

Rate(out of 10)	Frequency	Cumulative Freq.
10	1	1
9	2	3
8	9	12
7	9	21
6	5	26
5	11	37
4	11	48
3	6	54
2	4	58
1	2	60

Future Work

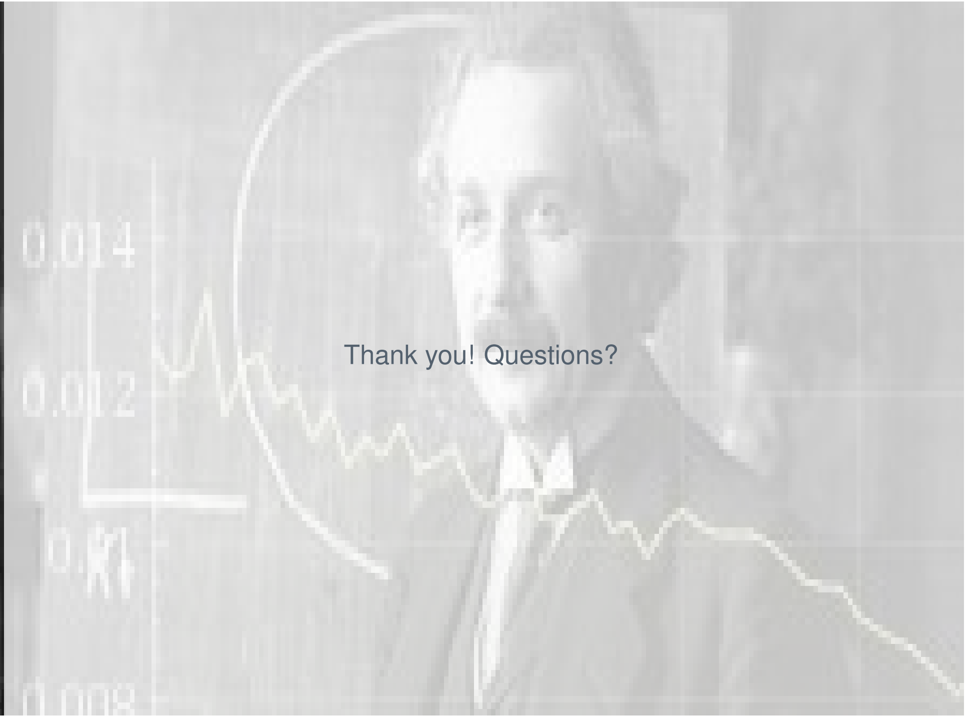
- Features for representation of parts are to be improved by incorporating visual features. Inclusion of visual features will also include the analysis of features like color, texture, etc. which is expected to improve the quality of evaluation.
- A feedback system can be added to the system as to increase edge weights to the features which are shopped together by users. This will be a self learning system and incorporate the changes in trending fashion all by itself.

References

[1] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar.
Rank aggregation methods for the web.
In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[2] G. Jeh and J. Widom.
Simrank: A measure of structural-context similarity.
In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[3] T. Zhu, P. Harrington, J. Li, and L. Tang.
Bundle recommendation in ecommerce.
In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 657–666, New York, NY, USA, 2014. ACM.



Thank you! Questions?